

[illegible]

Sy

LI

LI  
LI

LI

LI  
LI

LI

LI  
LILI  
LILI  
LILI  
LI

LI

LI  
LILI  
LI

11

LI  
LI

LI

LI  
LILI  
LI

54

LI

LI

1998, 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022, 2023, 2024, 2025, 2026, 2027, 2028, 2029, 2030, 2031, 2032, 2033, 2034, 2035, 2036, 2037, 2038, 2039, 2040, 2041, 2042, 2043, 2044, 2045, 2046, 2047, 2048, 2049, 2050, 2051, 2052, 2053, 2054, 2055, 2056, 2057, 2058, 2059, 2060, 2061, 2062, 2063, 2064, 2065, 2066, 2067, 2068, 2069, 2070, 2071, 2072, 2073, 2074, 2075, 2076, 2077, 2078, 2079, 2080, 2081, 2082, 2083, 2084, 2085, 2086, 2087, 2088, 2089, 2090, 2091, 2092, 2093, 2094, 2095, 2096, 2097, 2098, 2099, 2100, 2101, 2102, 2103, 2104, 2105, 2106, 2107, 2108, 2109, 2110, 2111, 2112, 2113, 2114, 2115, 2116, 2117, 2118, 2119, 2120, 2121, 2122, 2123, 2124, 2125, 2126, 2127, 2128, 2129, 2130, 2131, 2132, 2133, 2134, 2135, 2136, 2137, 2138, 2139, 2140, 2141, 2142, 2143, 2144, 2145, 2146, 2147, 2148, 2149, 2150, 2151, 2152, 2153, 2154, 2155, 2156, 2157, 2158, 2159, 2160, 2161, 2162, 2163, 2164, 2165, 2166, 2167, 2168, 2169, 2170, 2171, 2172, 2173, 2174, 2175, 2176, 2177, 2178, 2179, 2180, 2181, 2182, 2183, 2184, 2185, 2186, 2187, 2188, 2189, 2190, 2191, 2192, 2193, 2194, 2195, 2196, 2197, 2198, 2199, 2200, 2201, 2202, 2203, 2204, 2205, 2206, 2207, 2208, 2209, 2210, 2211, 2212, 2213, 2214, 2215, 2216, 2217, 2218, 2219, 2220, 2221, 2222, 2223, 2224, 2225, 2226, 2227, 2228, 2229, 2230, 2231, 2232, 2233, 2234, 2235, 2236, 2237, 2238, 2239, 2240, 2241, 2242, 2243, 2244, 2245, 2246, 2247, 2248, 2249, 2250, 2251, 2252, 2253, 2254, 2255, 2256, 2257, 2258, 2259, 2260, 2261, 2262, 2263, 2264, 2265, 2266, 2267, 2268, 2269, 2270, 2271, 2272, 2273, 2274, 2275, 2276, 2277, 2278, 2279, 2280, 2281, 2282, 2283, 2284, 2285, 2286, 2287, 2288, 2289, 2290, 2291, 2292, 2293, 2294, 2295, 2296, 2297, 2298, 2299, 2300, 2301, 2302, 2303, 2304, 2305, 2306, 2307, 2308, 2309, 2310, 2311, 2312, 2313, 2314, 2315, 2316, 2317, 2318, 2319, 2320, 2321, 2322, 2323, 2324, 2325, 2326, 2327, 2328, 2329, 2330, 2331, 2332, 2333, 2334, 2335, 2336, 2337, 2338, 2339, 2340, 2341, 2342, 2343, 2344, 2345, 2346, 2347, 2348, 2349, 2350, 2351, 2352, 2353, 2354, 2355, 2356, 2357, 2358, 2359, 2360, 2361, 2362, 2363, 2364, 2365, 2366, 2367, 2368, 2369, 2370, 2371, 2372, 2373, 2374, 2375, 2376, 2377, 2378, 2379, 2380, 2381, 2382, 2383, 2384, 2385, 2386, 2387, 2388, 2389, 2390, 2391, 2392, 2393, 2394, 2395, 2396, 2397, 2398, 2399, 2400, 2401, 2402, 2403, 2404, 2405, 2406, 2407, 2408, 2409, 2410, 2411, 2412, 2413, 2414, 2415, 2416, 2417, 2418, 2419, 2420, 2421, 2422, 2423, 2424, 2425, 2426, 2427, 2428, 2429, 2430, 2431, 2432, 2433, 2434, 2435, 2436, 2437, 2438, 2439, 2440, 2441, 2442, 2443, 2444, 2445, 2446, 2447, 2448, 2449, 2450, 2451, 2452, 2453, 2454, 2455, 2456, 2457, 2458, 2459, 2460, 2461, 2462, 2463, 2464, 2465, 2466, 2467, 2468, 2469, 2470, 2471, 2472, 2473, 2474, 2475, 2476, 2477, 2478, 2479, 2480, 2481, 2482, 2483, 2484, 2485, 2486, 2487, 2488, 2489, 2490, 2491, 2492, 2493, 2494, 2495, 2496, 2497, 2498, 2499, 2500, 2501, 2502, 2503, 2504, 2505, 2506, 2507, 2508, 2509, 2510, 2511, 2512, 2513, 2514, 2515, 2516, 2517, 2518, 2519, 2520, 2521, 2522, 2523, 2524, 2525, 2526, 2527, 2528, 2529, 2530, 2531, 2532, 2533, 2534, 2535, 2536, 2537, 2538, 2539, 2540, 2541, 2542, 2543, 2544, 2545, 2546, 2547, 2548, 2549, 2550, 2551, 2552, 2553, 2554, 2555, 2556, 2557, 2558, 2559, 2560, 2561, 2562, 2563, 2564, 2565, 2566, 2567, 2568, 2569, 2570, 2571, 2572, 2573, 2574, 2575, 2576, 2577, 2578, 2579, 2580, 2581, 2582, 2583, 2584, 2585, 2586, 2587, 2588, 2589, 2590, 2591, 2592, 2593, 2594, 2595, 2596, 2597, 2598, 2599, 2600, 2601, 2602, 2603, 2604, 2605, 2606, 2607, 2608, 2609, 2610, 2611, 2612, 2613, 2614, 2615, 2616, 2617, 2618, 2619, 2620, 2621, 2622, 2623, 2624, 2625, 2626, 2627, 2628, 2629, 2630, 2631, 2632, 2633, 2634, 2635, 2636, 2637, 2638, 2639, 2640, 2641, 2642, 2643, 2644, 2645, 2646, 2647, 2648, 2649, 2650, 2651, 2652, 2653, 2654, 2655, 2656, 2657, 2658, 2659, 2660, 2661, 2662, 2663, 2664, 2665, 2666, 2667, 2668, 2669, 2670, 2671, 2672, 2673, 2674, 2675, 2676, 2677, 2678, 2679, 26

10

10

11

LI

LI  
LILI  
LI

100

10

100

10

10

10

1

1

10

```

LL          IIIIII          SSSSSSSS
LL          IIIIII          SSSSSSSS
LL          II             SS
LL          II             SS
LL          II             SS
LL          II             SS
LL          II             SSSSSS
LL          II             SSSSSS
LL          II             SS
LL          II             SS
LL          II             SS
LL          II             SS
LL          II             SSSSSS
LLLLLLLLLLLL IIIIII          SSSSSSSS
LLLLLLLLLLLL IIIIII          SSSSSSSS

```



(2)	70	DECLARATIONS
(3)	143	OTSSGET1_DD Allocate a dynamic string
(4)	195	OTSSGET1_DD_R6 Allocate a dynamic string
(5)	246	OTSSFREE1_DD Deallocate a dynamic string
(6)	291	OTSSFREE1_DD6 Deallocate a dynamic string
(7)	334	OTSSFREE1_DD Deallocate N Dynamic Strings
(8)	383	OTSSFREE1_DD6 Deallocate N Dynamic Strings
(9)	425	OTSSCOPY_DXDX Copy String by Descriptor
(10)	519	OTSSCOPY_DXDX6 Copy String by Descriptor
(11)	667	OTSSCOPY_R_DX Copy String by Reference
(12)	746	OTSSCOPY_R_DX6 Copy Strings by reference
(13)	879	CHECK_FOR_FATAL_ERROR



```
0000 1 .TITLE OTSS$COPY - String copying module
0000 2 .IDENT /1-011/ ; File: OTSS$COPY.MAR Edit: SBL1011
0000 3
0000 4 :
0000 5 :*****
0000 6 :
0000 7 :* COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 8 :* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 9 :* ALL RIGHTS RESERVED.
0000 10 :
0000 11 :* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 12 :* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 13 :* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 14 :* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 15 :* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 16 :* TRANSFERRED.
0000 17 :
0000 18 :* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 19 :* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 20 :* CORPORATION.
0000 21 :
0000 22 :* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 23 :* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 24 :
0000 25 :
0000 26 :*****
0000 27 :
0000 28 :
0000 29 : FACILITY: Language-independent support: string handling
0000 30 :
0000 31 : ABSTRACT:
0000 32 :
0000 33 : This module contains routines to allocate and deallocate
0000 34 : strings. These entry points were in VMS release 1, before
0000 35 : there was a separate string facility, and they are being
0000 36 : retained for compatibility. They are implemented by calling
0000 37 : LIB$SGET1 DD R6, LIB$SFREE1 DD6 and LIB$SFREEN DD6.
0000 38 : This module also contains the routines to do string copying
0000 39 : using OTS$ semantics. They are implemented by calling
0000 40 : LIB$SCOPY_DXDX6 and LIB$SCOPY_R_DX6.
0000 41 :
0000 42 :
0000 43 : ENVIRONMENT: VAX-11 User Mode
0000 44 :
0000 45 : AUTHOR: R. Reichert, CREATION DATE: 3-APR-1981
0000 46 :
0000 47 : MODIFIED BY:
0000 48 :
0000 49 : 1-001 - Original. Based on Version 1-007 of OTSS$COPY.B32. With the
0000 50 : addition of the code to accomodate additional classes of
0000 51 : descriptors, necessitating a call to LIB$ANALYZE_SDESC_R3
0000 52 : it became increasingly difficult to control the register
0000 53 : usage in OTS$SCOPY_DXDX6 and OTS$SCOPY_R_DX6.
0000 54 : (In fact the original .B32 didn't control them correctly.)
0000 55 : RKR 3-APR-1981
0000 56 : 1-002 - Revise which error statuses get turned into signals in
0000 57 : CHECK_FOR_FATAL. RKR 3-SEP-1981
```



OTSS\$COPY  
1-011

- String copying module

D 3

16-SEP-1984 00:33:11 VAX/VMS Macro V04-00  
6-SEP-1984 11:15:27 [LIBRTL.SRC]OTSS\$COPY.MAR;1

Page 2  
(1)

```
0000 58 : 1-008 - Original OTSS$COPY.B32 had a revision history that ran up
0000 59 : through 1-007. To avoid confusion with module idents that
0000 60 : are out in the field, this module's ident must be at
0000 61 : least 1-008. RKR 14-SEP-1981
0000 62 : 1-009 - Add special-case code to process string descriptors that
0000 63 : "read" like fixed string descriptors. RKR 7-OCT-1981.
0000 64 : 1-010 - Redirect jsb's from LIB$ANALYZE_SDESC_R3 to
0000 65 : LIB$ANALYZE_SDESC_R2. RKR 18-NOV-1981.
0000 66 : 1-011 - Use general mode addressing. SBL 30-Nov-1981
0000 67 :
0000 68 :--
```



```
0000 70      .SBTTL  DECLARATIONS
0000 71  :
0000 72  : LIBRARY MACRO CALLS:
0000 73  :
0000 74      $$$DEF      : SS$ symbols
0000 75      $DSCDEF     : DSC$ symbols
0000 76  :
0000 77  : EXTERNAL DECLARATIONS:
0000 78  :
0000 79  : Prevent undeclared symbols from being automatically global.
0000 80  :
0000 81      .DSABL  GBL
0000 82  : The condition codes and signals we deal with
0000 83  :
0000 84      .EXTRN  OTSS_FATINTERR      ; Fatal internal error
0000 85      .EXTRN  LIB$_FATERRLIB     ;
0000 86  :
0000 87      .EXTRN  OTSS_INVSTRDES     ; Invalid string descriptor
0000 88      .EXTRN  LIB$_INVSTRDES     ;
0000 89  :
0000 90      .EXTRN  OTSS_INSVIRMEM     ; Insufficient virtual memory
0000 91      .EXTRN  LIB$_INSVIRMEM     ;
0000 92  :
0000 93      .EXTRN  LIB$_INVARG       ; Invalid argument
0000 94  :
0000 95      .EXTRN  LIB$_WRONUMARG    ; Wrong number of arguments
0000 96      .EXTRN  OTSS_WRONUMARG    ;
0000 97  :
0000 98  : The external routines we use
0000 99  :
0000 100     .EXTRN  LIB$STOP           ; Signal a fatal error
0000 101     .EXTRN  LIB$SGET1_DD_R6   ; Alloc. string by descr
0000 102     .EXTRN  LIB$SFREET_DD6     ; Free 1 by descr.
0000 103     .EXTRN  LIB$SFREEN_DD6    ; Free N by descr.
0000 104     .EXTRN  LIB$SCOPY_DXDX6    ; Copy string by descr
0000 105     .EXTRN  LIB$SCOPY_R_DX6    ; Copy string by ref.
0000 106     .EXTRN  LIB$ANALYZE_SDESC_R2 ; Analyze desc to get length and
0000 107     : address of data
0000 108  :
0000 109  : MACROS:
0000 110  :
0000 111     .MACRO  SIGNAL_FATAL_ERR ?L1
0000 112  :+
0000 113  : This macro checks to see if the current status in R0 is a success.
0000 114  : If so, it continues. If it is not a success, it branches to
0000 115  : CHECK_FOR_FATAL_ERROR for a closer look at the error code. If it
0000 116  : it is found to be one of a set of fatal errors of interest, the
0000 117  : corresponding OTSS error is signaled. Else the supplied error code
0000 118  : is signaled.
0000 119  :
0000 120     BLBS    R0, L1      ; If success code, bypass checks
0000 121     BRW     CHECK_FOR_FATAL_ERROR ; see if it is one of interest
0000 122  L1:
0000 123     .ENDM  SIGNAL_FATAL_ERR
0000 124  :
0000 125  :
0000 126  : EQUATED SYMBOLS:
```



```

0000 127 :
0000 128 :      NONE
0000 129 :
0000 130 :      OWN STORAGE:
0000 131 :
00000000 132 :      .PSECT _OTSS$DATA PIC, USR, CON, REL, LCL, NOSHR, -
0000 133 :      NOEXE, RD, WRT, LONG
0000 134 :
0000 135 :      NONE
0000 136 :
0000 137 :      PSECT DECLARATIONS:
0000 138 :
00000000 139 :      .PSECT _OTSS$CODE PIC, USR, CON, REL, LCL, SHR, -
00000000 140 :      EXE, RD, NOWRT, LONG
0000 141 :

```

```
0000 143 .SBTTL OTSS$GET1_DD Allocate a dynamic string
0000 144 :++
0000 145 : FUNCTIONAL DESCRIPTION:
0000 146 :
0000 147 : Allocate a string. LEN bytes are allocated to DESCRIP, which
0000 148 : is assumed to be a dynamic descriptor. If the descriptor
0000 149 : already has storage allocated to it, but not enough, the old
0000 150 : storage is deallocated.
0000 151 :
0000 152 : CALLING SEQUENCE:
0000 153 :
0000 154 : status.wlc.v = OTSS$GET1_DD (LEN.rwu.v, DESCRIP.wqu.r)
0000 155 :
0000 156 : FORMAL PARAMETERS:
0000 157 :
0000 158 : LEN.rwu.v The number of bytes to allocate.
0000 159 : DESCRIP.wqu.r The descriptor. The DSC$B_DTYPE field is not
0000 160 : touched.
0000 161 :
0000 162 : IMPLICIT INPUTS:
0000 163 :
0000 164 : NONE
0000 165 :
0000 166 : IMPLICIT OUTPUTS:
0000 167 :
0000 168 : NONE
0000 169 :
0000 170 : ROUTINE VALUE:
0000 171 : COMPLETION CODES:
0000 172 :
0000 173 : NONE
0000 174 :
0000 175 : SIDE EFFECTS:
0000 176 :
0000 177 : May deallocate the descriptor's storage and allocate new
0000 178 : storage for it.
0000 179 : May signal OTSS$_INSVIRMEM, OTSS$_FATINTERR
0000 180 :
0000 181 :--
0000 182 :
0000 183 : Displacements from AP
0000 184 :
00000004 0000 185 LEN = 4
00000008 0000 186 DESCRIP = 8
0000 187
007C 0000 188 .ENTRY OTSS$GET1_DD, ^M<R2,R3,R4,R5,R6> ; Entry point
50 04 AC 3C 0002 189 MOVZWL LEN(AP), R0 ; length desired to R0
51 08 AC D0 0006 190 MOVL DESCRIP(AP), R1 ; descriptor address to R1
00000000'GF 16 000A 191 JSB G^LIB$GET1_DD_R6 ; go allocate
0010 192 SIGNAL_FATAL_ERR ; signal if a fatal error
04 0016 193 RET ; to caller
```



```
0017 195 .SBTTL OTSS$GET1_DD_R6 Allocate a dynamic string
0017 196 :++
0017 197 : FUNCTIONAL DESCRIPTION:
0017 198 :
0017 199 :     Allocate a string.  LEN bytes are allocated to DESCRIP, which
0017 200 :     is assumed to be a dynamic descriptor.  If the descriptor
0017 201 :     already has storage allocated to it, but not enough, the old
0017 202 :     storage is deallocated.
0017 203 :
0017 204 : CALLING SEQUENCE
0017 205 :
0017 206 :     status.wlc.v = JSB OTSS$GET1_DD_R6 (LEN.rwu.v, DESCRIP.wqu.r)
0017 207 :
0017 208 : FORMAL PARAMETERS:
0017 209 :
0017 210 :     LEN.rwu.v      In R0, the number of bytes to allocate.
0017 211 :     DESCRIP.wqu.r  In R1, The descriptor.  The DSC$B_DTYPE field
0017 212 :                   is not touched.
0017 213 :
0017 214 : IMPLICIT INPUTS:
0017 215 :
0017 216 :     NONE
0017 217 :
0017 218 : IMPLICIT OUTPUTS:
0017 219 :
0017 220 :     NONE
0017 221 :
0017 222 : ROUTINE VALUE:
0017 223 : COMPLETION CODES:
0017 224 :
0017 225 :     NONE
0017 226 :
0017 227 : SIDE EFFECTS:
0017 228 :
0017 229 :     May deallocate the descriptor's storage and allocate new
0017 230 :     storage for it.
0017 231 :     May signal OTSS$_INSVIRMEM or OTSS$_FATINTERR
0017 232 :
0017 233 : --
0017 234 :
0017 235 OTSS$GET1_DD_R6::
0017 236
0017 237     50    50    3C    MOVZWL  R0, R0                ; extract words worth of length
001A 238
001A 239                ; R1 already contains address of
001A 240                ; descriptor
001A 241
001A 242     00000000'GF    16    JSB      G^LIB$GET1_DD_R6    ; go allocate
0020 243                ; signal error if a fatal one
0026 244     05    0026    RSB                        ; return to our caller
```

```

0027 246 .SBTTL OTSS$FREE1_DD Deallocate a dynamic string
0027 247 :++
0027 248 : FUNCTIONAL DESCRIPTION:
0027 249 :
0027 250 :     Deallocate a string. The string is assumed to be dynamic.
0027 251 :     If it isn't, LIBSSFREE1_DD6 will take care of it.
0027 252 :
0027 253 : CALLING SEQUENCE:
0027 254 :
0027 255 :     status.wlc.v = OTSS$FREE1_DD (DESCRIP.wqu.r)
0027 256 :
0027 257 : FORMAL PARAMETERS:
0027 258 :
0027 259 :     DESCRIP.wqu.r   The descriptor of the string to deallocate.
0027 260 :
0027 261 : IMPLICIT INPUTS:
0027 262 :
0027 263 :     NONE
0027 264 :
0027 265 : IMPLICIT OUTPUTS:
0027 266 :
0027 267 :     NONE
0027 268 :
0027 269 : ROUTINE VALUE:
0027 270 : COMPLETION CODES:
0027 271 :
0027 272 :     NONE
0027 273 :
0027 274 : SIDE EFFECTS:
0027 275 :
0027 276 :     May deallocate virtual storage.
0027 277 :     May signal OTSS$_FATINTERR
0027 278 :
0027 279 :--
0027 280 :
0027 281 : Displacements from AP
0027 282 :
00000004 0027 283 DESCRIP =      4
0027 284 :
007C 0027 285 .ENTRY OTSS$FREE1_DD, ^M<R2,R3,R4,R5,R6> ; Entry point
50 04 AC D0 0029 286 MOVL DESCRIP(AP), R0 ; address of descriptor to R0
00000000'GF 16 002D 287 JSB G^LIBSSFREE1_DD6 ; go free string
0033 288 SIGNAL_FATAL_ERR ; signal if error is fatal
04 0039 289 RET ; to caller
  
```



```

003A 291 .SBTTL OTSS$FREE1_DD6 Deallocate a dynamic string
003A 292 :++
003A 293 : FUNCTIONAL DESCRIPTION:
003A 294 :
003A 295 : Deallocate a string. The string is assumed to be dynamic.
003A 296 : If it isn't, LIB$FREE1_DD6 will take care of it.
003A 297 :
003A 298 : CALLING SEQUENCE:
003A 299 :
003A 300 : status.wlc.v = JSB OTSS$FREE1_DD6 (DESCRIP.wqu.r)
003A 301 :
003A 302 : FORMAL PARAMETERS:
003A 303 :
003A 304 : DESCRIP.wqu.r In R0, the descriptor of the string to
003A 305 : deallocate.
003A 306 :
003A 307 : IMPLICIT INPUTS:
003A 308 :
003A 309 : NONE
003A 310 :
003A 311 : IMPLICIT OUTPUTS:
003A 312 :
003A 313 : NONE
003A 314 :
003A 315 : ROUTINE VALUE:
003A 316 : COMPLETION CODES:
003A 317 :
003A 318 : NONE
003A 319 :
003A 320 : SIDE EFFECTS:
003A 321 :
003A 322 : May deallocate virtual storage.
003A 323 : May signal OTSS$_FATINTERR
003A 324 :
003A 325 : --
003A 326 :
003A 327 OTSS$FREE1_DD6::
003A 328
003A 329 ; R0 already contains address of
003A 330 ; descriptor to be freed
003A 331 ; go free string
003A 332 ; check for fatal error
003A 333 ; return to our caller
00000000'GF 16 003A 330 JSB G^LIB$FREE1_DD6
0040 331 SIGNAL_FATAL_ERR
05 0046 332 RSB

```

```

0047 334 .SBTTL OTSS$FREEEN_DD Deallocate N Dynamic Strings
0047 335 :++
0047 336 : FUNCTIONAL DESCRIPTION:
0047 337 :
0047 338 : Deallocate a number of strings. The strings are all assumed
0047 339 : to be dynamic. If not, LIB$FREE1_DD6 will eventually take care
0047 340 : of them.
0047 341 :
0047 342 : CALLING SEQUENCE:
0047 343 :
0047 344 : status.wlc.v = OTSS$FREEEN_DD (NUM_DESC.rwu.v, DESC_PTR.wqu.r)
0047 345 :
0047 346 : FORMAL PARAMETERS:
0047 347 :
0047 348 : NUM_DESC.rwu.v The number of descriptors to deallocate.
0047 349 : DESC_PTR.wqu.r The first of these descriptors.
0047 350 :
0047 351 : IMPLICIT INPUTS:
0047 352 :
0047 353 : NONE
0047 354 :
0047 355 : IMPLICIT OUTPUTS:
0047 356 :
0047 357 : NONE
0047 358 :
0047 359 : ROUTINE VALUE:
0047 360 : COMPLETION CODES:
0047 361 :
0047 362 : SSS_NORMAL
0047 363 :
0047 364 : SIDE EFFECTS:
0047 365 :
0047 366 : May deallocate virtual storage.
0047 367 :
0047 368 :--
0047 369 :
0047 370 :
0047 371 : Displacements from AP
0047 372 :
00000004 0047 373 NUM_DESC = 4
00000008 0047 374 DESC_PTR = 8
0047 375 :
0047 376 .ENTRY OTSS$FREEEN_DD, ^M<R2,R3,R4,R5,R6> ; Entry point
50 04 AC 007C 0049 377 MOVQ NUM_DESC(AP), R0 ; number of desc ==> R0
004D 378 ; address of first desc ==> R1
00000000'GF 16 004D 379 JSB G^LIB$FREEEN_DD6 ; go free N descriptors
0053 380 SIGNAL_FATAL_ERR ; check for fatal error
04 0059 381 RET ; to caller

```



```

005A 383 .SBTTL OTSS$SFREEN_DD6 Deallocate N Dynamic Strings
005A 384 :++
005A 385 : FUNCTIONAL DESCRIPTION:
005A 386 :
005A 387 : Deallocate a number of strings. The strings are all assumed
005A 388 : to be dynamic. If they aren't, eventually LIB$SFREEN1_DD6 will
005A 389 : take care of them.
005A 390 :
005A 391 : CALLING SEQUENCE:
005A 392 :
005A 393 : status.wlc.v = JSB OTSS$SFREEN_DD6 (NUM_DESC.rl.v, DESC_PTR.wqu.r)
005A 394 :
005A 395 : FORMAL PARAMETERS:
005A 396 :
005A 397 : NUM_DESC.rl.v In R0, the number of descriptors to deallocate.
005A 398 : DESC_PTR.wqu.r In R1, the address of first of these descriptors
005A 399 :
005A 400 : IMPLICIT INPUTS:
005A 401 :
005A 402 : NONE
005A 403 :
005A 404 : IMPLICIT OUTPUTS:
005A 405 :
005A 406 : NONE
005A 407 :
005A 408 : ROUTINE VALUE:
005A 409 : COMPLETION CODES:
005A 410 :
005A 411 : SS$_NORMAL
005A 412 :
005A 413 : SIDE EFFECTS:
005A 414 :
005A 415 : May deallocate virtual storage.
005A 416 : May signal OTSS$_FATINTERR
005A 417 :
005A 418 : --
005A 419 :
005A 420 OTSS$SFREEN DD6::
005A 421 JSB G^LIB$SFREEN_DD6 ; let LIB$SFREEN_DD6 do it
0060 422 SIGNAL_FATAL_ERR ; check for fatal error
05 0066 423 RSB ; return to caller.

```

```

0067 425 .SBTTL OTSS$COPY_DXDX Copy String by Descriptor
0067 426 :++
0067 427 : FUNCTIONAL DESCRIPTION:
0067 428 :
0067 429 :     Copy any supported class string passed by descriptor to any
0067 430 :     supported class string.
0067 431 :
0067 432 : CALLING SEQUENCE:
0067 433 :
0067 434 :     status.wlc.v = OTSS$COPY_DXDX (SRC_DESC.rt.dx, DEST_DESC.wt.dx)
0067 435 :
0067 436 : FORMAL PARAMETERS:
0067 437 :
0067 438 :     SRC_DESC.rt.dx The source descriptor.
0067 439 :     DEST_DESC.wt.dx The destination descriptor. The class and dtype
0067 440 :                     fields are not disturbed.
0067 441 :
0067 442 : IMPLICIT INPUTS:
0067 443 :
0067 444 :     NONE
0067 445 :
0067 446 : IMPLICIT OUTPUTS:
0067 447 :
0067 448 :     NONE
0067 449 :
0067 450 : ROUTINE VALUE:
0067 451 : COMPLETION CODES:
0067 452 :
0067 453 :     The number of bytes of the source not moved to the destination.
0067 454 :
0067 455 : SIDE EFFECTS:
0067 456 :
0067 457 :     May allocate and deallocate virtual storage.
0067 458 :     May signal OTSS$_INVSTRDES, OTSS$_INSVIRMEM, or OTSS$_FATINTERR.
0067 459 :
0067 460 :--
0067 461 :
0067 462 : Displacements from AP
0067 463 :
00000004 0067 464 SRC_DESC = 4
00000008 0067 465 DEST_DESC = 8
0067 466 :
007C 0067 467 .ENTRY OTSS$COPY_DXDX, ^M<R2,R3,R4,R5,R6> ; Entry point
0069 468 :+
0069 469 : Copy string using LIB$COPY_DXDX6
0069 470 :-
0069 471 : MOVQ SRC_DESC(AP), R0 ; load R0 and R1 with addresses
006D 472 : ; source and destination
006D 473 : ; descriptors
006D 474 : JSB G^LIB$COPY_DXDX6 ; go copy string
0073 475 : SIGNAL_FATAL_ERR ; check for fatal error
0079 476 :
0079 477 :+
0079 478 : Compute length of source string and save it in R4
0079 479 : (no need to check status after call to LIB$ANALYZE_SDESC R2. If
0079 480 : there was anything wrong with source descriptor, LIB$COPY_DXDX6
0079 481 : would already have complained about it.)

```



```

50 04 AC D0 0079 482 :-
02 03 A0 91 0079 483      MOVL SRC_DESC(AP), R0      ; address of source descr.
      05 1A 007D 484      CMPB DSC$B_CLASS(R0), #DSC$K_CLASS_D ; read like fixed ?
      54 60 3C 0081 485      BGTRU 1$                ; no
      09 11 0083 486      MOVZWL DSC$W_LENGTH(R0), R4    ; length -> R4
      11 0086 487      BRB 2$                ; join common flow
00000000'GF 16 0088 488
      54 51 3C 008E 489 1$: JSB G^LIB$ANALYZE_SDESC_R2    ; extract length of source
      0091 491      MOVZWL R1, R4                ; length of source string
      0091 492      :-
      0091 493      ;+
      0091 494      ; Compute length of destination string
      0091 495      ; (no need to check status after call to LIB$ANALYZE_SDESC_R2. If
      0091 496      ; there was anything wrong with destination descriptor, LIB$COPY_DXDX6
      0091 497      ; would already have complained about it.)
      0091 498 2$: MOVL DEST_DESC(AP), R0      ; address of destination descr
      02 03 A0 91 0095 499      CMPB DSC$B_CLASS(R0), #DSC$K_CLASS_D ; read like fixed ?
      51 09 1A 0099 500      BGTRU 3$                ; no
      52 04 A0 D0 009B 501      MOVZWL DSC$W_LENGTH(R0), R1    ; length -> R1
      06 11 009E 502      MOVL DSC$A_POINTER(R0), R2        ; address -> R2
      00A2 503      BRB 4$                ; join common flow
00000000'GF 16 00A4 504
      00AA 505 3$: JSB G^LIB$ANALYZE_SDESC_R2    ; extract length of destination
      00AA 506      :-
      00AA 507      ;+
      00AA 508      ; At this point, R1 is length of destination, R2 is address of
      00AA 509      ; destination, and R4 is length of source.
      00AA 510      ; Compute MAX (0, source length - destination_length). This becomes
      00AA 511      ; the number of unmoved bytes which must end up in R0.
      00AA 512      :-
50 54 51 C3 00AA 513 4$: SUBL3 R1, R4, R0          ; source len - destination len
      02 18 00AE 514      BGEQ 5$                ; if positive
      50 D4 00B0 515      CLRL R0                ; else zero
      00B2 516 5$:
      04 00B2 517      RET
```

```

00B3 519      .SBTTL  OTSS$SCOPY_DXDX6 Copy String by Descriptor
00B3 520      :++
00B3 521      : FUNCTIONAL DESCRIPTION:
00B3 522      :
00B3 523      :     Copy any supported class string passed by descriptor to any
00B3 524      :     supported class string.
00B3 525      :
00B3 526      : CALLING SEQUENCE:
00B3 527      :
00B3 528      :     status.wlc.v = JSB OTSS$SCOPY_DXDX6 (SRC_DESC.rt.dx,
00B3 529      :                                     DEST_DESC.wt.dx)
00B3 530      :
00B3 531      : FORMAL PARAMETERS:
00B3 532      :
00B3 533      :     SRC_DESC.rt.dx  The source descriptor, in R0.
00B3 534      :     DEST_DESC.wt.dx The destination descriptor. The class and dtype
00B3 535      :                     fields are not disturbed. This is in R1.
00B3 536      :
00B3 537      : IMPLICIT INPUTS:
00B3 538      :
00B3 539      :     None
00B3 540      :
00B3 541      : IMPLICIT OUTPUTS:
00B3 542      :
00B3 543      :     R0      Number of unmoved bytes remaining in source
00B3 544      :           string.
00B3 545      :     R1      Address one byte beyond the last byte in the
00B3 546      :           source string that was moved.
00B3 547      :     R2      0
00B3 548      :     R3      Address one byte beyond the destination string
00B3 549      :     R4      0
00B3 550      :     R5      0
00B3 551      :     PSL<N>  1 = Source length less than destination length
00B3 552      :     PSL<Z>  1 = Source length equals destination length
00B3 553      :     PSL<V>  0
00B3 554      :     PSL<C>  1 = Source length LESS destination length
00B3 555      :
00B3 556      : ROUTINE VALUE:
00B3 557      : COMPLETION CODES:
00B3 558      :
00B3 559      :     See IMPLICIT OUTPUTS, above.
00B3 560      :
00B3 561      : SIDE EFFECTS:
00B3 562      :
00B3 563      :     May allocate and deallocate virtual storage.
00B3 564      :     May signal OTSS$_INVSTRDES, OTSS$_INSVIRMEM, OTSS$_FATINTERR
00B3 565      :
00B3 566      : --
00B3 567      :
00B3 568      : Temp locations on stack
00B3 569      :
00000000 00B3 570 TEMP_SRC_ADDR  = 0
00000004 00B3 571 TEMP_DST_ADDR  = 4
00000008 00B3 572 TEMP_SRC_LEN   = 8
0000000C 00B3 573 TEMP_DST_LEN   = 12
00000010 00B3 574 STACK_SPACE  = 16
00B3 575

```



```
00B3 576 OTSS$COPY_DXDX6::
00B3 577 ;+
00B3 578 ; Save R0 (source desc addr) and R1 (dest descr addr) on the stack.
00B3 579 ;+
00B3 580 ;
00B3 581 ;
00B3 582 ;
00B3 583 ;+
00B3 584 ; Copy string using LIB$SCOPY_DXDX6
00B3 585 ;+
00B3 586 ;
00B3 587 ;
00B3 588 ;+
00B3 589 ; Compute length and address of source string and save on stack
00B3 590 ; (no need to check status after call to LIB$ANALYZE_SDESC_R2. If
00B3 591 ; there was anything wrong with source descriptor, [IB$SCOPY_DXDX6
00B3 592 ; would already have complained about it.)
00B3 593 ;+
00B3 594 ;
00B3 595 ;
00B3 596 ;
00B3 597 ;
00B3 598 ;
00B3 599 ;
00B3 600 ;
00B3 601 1$: JSB G^LIB$ANALYZE_SDESC_R2 ; extract length of source
00B3 602 ; length of source string
00B3 603 ; addr of 1st byte of source
00B3 604 ;+
00B3 605 ;
00B3 606 ; Compute length and address of destination string and save on stack
00B3 607 ; (no need to check status after call to LIB$ANALYZE_SDESC_R2. If
00B3 608 ; there was anything wrong with destination descriptor, LIB$SCOPY_DXDX6
00B3 609 ; would already have complained about it.)
00B3 610 ;+
00B3 611 2$: MOVL TEMP_DST_ADDR(SP), R0 ; address of destination descr
00B3 612 ; read like fixed ?
00B3 613 ;
00B3 614 ;
00B3 615 ;
00B3 616 ;
00B3 617 ;
00B3 618 3$: JSB G^LIB$ANALYZE_SDESC_R2 ; extract length of destination
00B3 619 ; length of dest string
00B3 620 ; address of 1st byte of dest.
00B3 621 ;
00B3 622 ; Compute MAX (0, source length - destination length). This becomes
00B3 623 ; the number of unmoved bytes which must eventually end up in R0.
00B3 624 ;+
00B3 625 4$: SUBL3 TEMP_DST_LEN(SP), TEMP_SRC_LEN(SP), R6
00B3 626 ; if positive
00B3 627 ; else zero
00B3 628 5$:
00B3 629 ;+
00B3 630 ;
00B3 631 ; Compute address of first unmoved source byte as
00B3 632 ; R1 = TEMP_SRC_ADDR + MIN (TEMP_SRC_LEN, TEMP_DST_LEN)
```

5E 10 C2  
6E 50 D0  
04 AE 51 D0

00000000'GF 16

50 6E D0  
02 03 A0 91  
0A 1A 00D0  
08 AE 60 3C 00D2  
6E 04 A0 D0  
0D 11 00DA  
00DC 600

00000000'GF 16  
08 AE 51 D0  
6E 52 D0

50 04 AE D0  
02 03 A0 91  
0B 1A 00F1  
0C AE 60 3C 00F3  
04 AE 04 A0 D0  
0E 11 00FC  
00FE 617

00000000'GF 16  
0C AE 51 D0  
04 AE 52 D0

56 08 AE 0C AE C3  
02 18 0112  
56 D4 0114  
0116 628  
0116 629  
0116 630  
0116 631  
0116 632



```

08 AE 0C AE D1 0116 633 :-
18 07 18 0116 634 CMPL TEMP_DST_LEN(SP), TEMP_SRC_LEN(SP)
51 0C AE 6E C1 011B 635 BGEQ 6$ ; Destination length bigger
11 05 11 011D 636 ADDL3 TEMP_SRC_ADDR(SP), TEMP_DST_LEN(SP), R1 ;dst len smaller
51 08 AE 6E C1 0122 637 BRB 7$
0124 638 6$: ADDL3 TEMP_SRC_ADDR(SP), TEMP_SRC_LEN(SP), R1 ;src len smaller
0129 639 7$:
0129 640
0129 641 ;+
0129 642 ; Compute address of one byte beyond last byte written to destination
0129 643 ; string as
0129 644 R3 = TEMP_DST_ADDR + TEMP_DST_LEN
0129 645 :-
53 0C AE 04 AE C1 0129 646 ADDL3 TEMP_DST_ADDR(SP), TEMP_DST_LEN(SP), R3
012F 647
012F 648 ;+
012F 649 ; Set up all remaining registers as specified under IMPLICIT OUTPUTS
012F 650 ; above.
012F 651
012F 652 ; Final movement must be into R0 in order to set up the condition codes
012F 653 ; correctly.
012F 654
012F 655 ; First, we restore the stack.
012F 656 :-
5E 10 C0 012F 657 ADDL2 #STACK_SPACE, SP ; restore stack address
0132 658
54 7C 0132 659 CLRQ R4 ; R4 and R5 are zero
0134 660 ; R3 all set
52 D4 0134 661 CLRL R2 ; R2 is zero
0136 662 ; R1 all set
50 56 D0 0136 663 MOVL R6, R0 ; Set R0 and set condition codes
0139 664 ; properly
05 0139 665 RSB ; return to caller

```



```

013A 667      .SBTTL OTSS$SCOPY_R_DX Copy String by Reference
013A 668      :++
013A 669      : FUNCTIONAL DESCRIPTION:
013A 670      :
013A 671      :     Copy any string passed by reference to any supported class
013A 672      : string passed by descriptor.
013A 673      :
013A 674      : CALLING SEQUENCE:
013A 675      :
013A 676      :     status.wlc.v = OTSS$SCOPY_R_DX (SRC_LEN.rwu.v, SRC_ADDR.rt.r,
013A 677      :                                     DEST_DESC.wt.dx)
013A 678      :
013A 679      : FORMAL PARAMETERS:
013A 680      :
013A 681      :     SRC_LEN.rwu.v   The number of bytes of data in the source
013A 682      :     SRC_ADDR.rt.r   The address of the first of those bytes.
013A 683      :     DEST_DESC.wt.dx  The destination descriptor.  The class and dtype
013A 684      :                     fields are not disturbed.
013A 685      :
013A 686      : IMPLICIT INPUTS:
013A 687      :
013A 688      :     NONE
013A 689      :
013A 690      : IMPLICIT OUTPUTS:
013A 691      :
013A 692      :     NONE
013A 693      :
013A 694      : ROUTINE VALUE:
013A 695      : COMPLETION CODES:
013A 696      :
013A 697      :     The number of unmoved source bytes, or 0 if there are no unmoved
013A 698      :     source bytes.
013A 699      :
013A 700      : SIDE EFFECTS:
013A 701      :
013A 702      :     May allocate and deallocate virtual storage.
013A 703      :     May signal OTSS$_INVSTRDES, OTSS$_INSVIRMEM, OTSS$_FATINTERR
013A 704      :
013A 705      : --
013A 706      :
013A 707      : Displacements off AP
013A 708      :
00000004 013A 709 SRC_LEN      = 4
00000008 013A 710 SRC_ADDR    = 8
0000000C 013A 711 DEST_DESC   = 12
013A 712      :
007C 013A 713      .ENTRY OTSS$SCOPY_R_DX, ^M<R2,R3,R4,R5,R6>      ; Entry point
013C 714      :+
013C 715      : Copy string using LIB$SCOPY_R_DX6
013C 716      : -
013C 717      :     MOVL SRC_LEN(AP), R0      ; R0 is length of source
013C 718      :     MOVQ SRC_ADDR(AP), R1     ; R1 is addr of source
013C 719      :                                     ; R2 is addr of dest desc
0144 720      :     JSB G^LIB$SCOPY_R_DX6 ; copy the string
0144 721      :     SIGNAL_FATAL_ERR          ; check for fatal error
0150 722      :
0150 723      :+
  
```

```

0150 724 : Compute length of destination string (into R1).
0150 725 : (no need to check status after call to LIB$ANALYZE_SDESC_R2. If
0150 726 : there was anything wrong with destination descriptor, LIB$COPY_DXDX6
0150 727 : would already have complained about it.)
0150 728 :-
50 0C AC D0 0150 729      MOVL    DEST_DESC(AP), R0      ; address of dest descr
02 03 A0 91 0154 730      CMPB    DSC$B_CLASS(R0), #DSC$K_CLASS_D ; rad like fixed ?
      05 1A 0158 731      BGTRU    1$              ; no
      51 60 3C 015A 732      MOVZWL  DSC$W_LENGTH(R0), R1      ; length -> R1
      06 11 015D 733      BRB       2$              ; join common flow
00000000'GF 16 015F 734      JSB      G^LIB$ANALYZE_SDESC_R2 ; get length
      0165 735 1$:
      0165 736 :+
      0165 737 : Compute R0 = MAX (0, source_len - destination_length)
      0165 738 :-
50 52 04 AC 3C 0165 739 2$:      MOVZWL  SRC_LEN(AP), R2      ; words worth of source length
50 52 51 C3 0169 740      SUBL3    R1, R2, R0      ; R0 = dest len - src len
      02 18 016D 741      BGEQ     3$              ; if positive
      50 D4 016F 742      CLRL     R0              ; else zero
      0171 743 3$:
04 0171 744      RET                      ; to caller

```



```

0172 746 .SBTTL OTSS$COPY_R_DX6 Copy Strings by reference
0172 747 :++
0172 748 : FUNCTIONAL DESCRIPTION:
0172 749 :
0172 750 :     Copy any class string passed by reference to any supported
0172 751 :     class string passed by descriptor.
0172 752 :
0172 753 : CALLING SEQUENCE:
0172 754 :
0172 755 :     status.wlc.v = JSB OTSS$COPY_R_DX6 (SRC_LEN.rwu.v, SRC_ADDR.rt.r,
0172 756 :                                     DEST_DESC.wt.dx)
0172 757 :
0172 758 : FORMAL PARAMETERS:
0172 759 :
0172 760 :     SRC_LEN.rwu.v    The number of source bytes, in R0.
0172 761 :     SRC_ADDR.rt.r    Address of the first of these bytes, in R1.
0172 762 :     DEST_DESC.wt.dx  The destination descriptor. The class and
0172 763 :                     dtype fields are not disturbed. This is in R2
0172 764 :
0172 765 : IMPLICIT INPUTS:
0172 766 :
0172 767 :     None
0172 768 :
0172 769 : IMPLICIT OUTPUTS:
0172 770 :
0172 771 :     R0    Number of unmoved bytes remaining in source
0172 772 :           string.
0172 773 :     R1    Address one byte beyond the last byte in the
0172 774 :           source string that was moved.
0172 775 :     R2    0
0172 776 :     R3    Address one byte beyond the destination string
0172 777 :           0
0172 778 :     R4    0
0172 779 :     R5    0
0172 780 :     PSL<N> 1 = Source length less than destination length
0172 781 :     PSL<Z> 1 = Source length equals destination length
0172 782 :     PSL<V> 0
0172 783 :     PSL<C> 1 = Source length LSSU destination length
0172 784 :
0172 785 : ROUTINE VALUE:
0172 786 : COMPLETION CODES:
0172 787 :
0172 788 :     See IMPLICIT OUTPUTS, above.
0172 789 :
0172 790 : SIDE EFFECTS:
0172 791 :
0172 792 :     May allocate and deallocate virtual storage.
0172 793 :     May signal OTSS$_INVSTRDES, OTSS$_INSVIRMEM, OTSS$_FATINTERR.
0172 794 : --
0172 795 :
0172 796 : Temp locations on stack
0172 797 :
00000000 0172 798 TEMP_SRC_ADDR = 0
00000004 0172 799 TEMP_DST_ADDR = 4
00000008 0172 800 TEMP_SRC_LEN = 8
0000000C 0172 801 TEMP_DST_LEN = 12
00000010 0172 802 STACK_SPACE = 16

```

```

0172 803
0172 804 OTSS$COPY_R_DX6::
0172 805 ;+
0172 806 ; Save input arguments (in R0 - R2) onto stack
0172 807 ; -
0172 808         SUBL2    #STACK SPACE, SP        ; make space on stack
08 AE 10 C2 0172 809         MOVZWL   R0, TEMP_SRC_LEN(SP)    ; Save source length
06 AE 50 3C 0175 810         MOVL     R1, TEMP_SRC_ADDR(SP)   ; Save source descr addr
04 AE 51 D0 0179 811         MOVL     R2, TEMP_DST_ADDR(SP)  ; Save destination descr addr
04 AE 52 D0 017C 812 ;+
0180 813 ; Copy string using LIB$SCOPY_R_DX6
0180 814 ; -
00000000'GF 16 0180 815         JSB      G^LIB$SCOPY_R_DX6        ; go copy string
0186 816         SIGNAL_FATAL_ERR                ; check for fatal error
018C 817 ;+
018C 818 ; Compute length and address of destination string and save on stack
018C 819 ; (no need to check status after call to LIB$ANALYZE_SDESC_R2. If
018C 820 ; there was anything wrong with destination descriptor, LIB$SCOPY_DXDX6
018C 821 ; would already have complained about it.)
018C 822 ; -
018C 823         MOVL     TEMP_DST_ADDR(SP), R0    ; address of destination descr
02 04 AE D0 018C 824         CMPB     DSC$B_CLASS(R0), #DSC$K_CLASS_D ; read like fixed ?
03 A0 91 0190 825         BGTRU     1$                ; no
0C AE 0B 1A 0194 826         MOVZWL   DSC$W_LENGTH(R0), TEMP_DST_LEN(SP) ; length
04 AE 04 A0 3C 0196 827         MOVL     DSC$A_POINTER(R0), TEMP_DST_ADDR(SP) ; length
0E 11 019F 828         BRB      2$                ; join common flow
01A1 829
00000000'GF 16 01A1 830 1$:      JSB      G^LIB$ANALYZE_SDESC_R2 ; extract length of destination
0C AE 51 D0 01A7 831         MOVL     R1, TEMP_DST_LEN(SP)    ; length of dest string
04 AE 52 D0 01AB 832         MOVL     R2, TEMP_DST_ADDR(SP)   ; address of 1st byte of dest.
01AF 833
01AF 834 ; Compute MAX (0, source length - destination length). This becomes
01AF 835 ; the number of unmoved bytes which must eventually end up in R0.
01AF 836 ; -
56 08 AE 0C AE C3 01AF 837 2$:      SUBL3     TEMP_DST_LEN(SP), TEMP_SRC_LEN(SP), R6
02 18 01B5 838         BGEQ      3$                ; if positive
56 56 D4 01B7 839         CLRL     R6                ; else zero
01B9 840 3$:
01B9 841
01B9 842 ;+
01B9 843 ; Compute address of first unmoved source byte as
01B9 844 ; R1 = TEMP_SRC_ADDR + MIN (TEMP_SRC_LEN, TEMP_DST_LEN)
01B9 845 ; -
08 AE 0C AE D1 01B9 846         CMPL     TEMP_DST_LEN(SP), TEMP_SRC_LEN(SP)
07 18 01BE 847         BGEQ      4$                ; Destination length bigger
51 0C AE 06 E C1 01C0 848         ADDL3     TEMP_SRC_ADDR(SP), TEMP_DST_LEN(SP), R1 ;dst len smaller
05 11 01C5 849         BRB      5$                ;
51 08 AE 06 E C1 01C7 850 4$:      ADDL3     TEMP_SRC_ADDR(SP), TEMP_SRC_LEN(SP), R1 ;src len smaller
01CC 851 5$:
01CC 852
01CC 853 ;+
01CC 854 ; Compute address of one byte beyond last byte written to destination
01CC 855 ; string as
01CC 856 ; R3 = TEMP_DST_ADDR + TEMP_DST_LEN
01CC 857 ; -
53 0C AE 04 AE C1 01CC 858         ADDL3     TEMP_DST_ADDR(SP), TEMP_DST_LEN(SP), R3
01D2 859

```



			01D2	860	:+		
			01D2	861	:	Set up all remaining registers as specified under IMPLICIT OUTPUTS	
			01D2	862	:	above.	
			01D2	863	:		
			01D2	864	:	Final movement must be into R0 in order to set up the condition codes	
			01D2	865	:	correctly.	
			01D2	866	:		
			01D2	867	:	First, we restore the stack.	
			01D2	868	:-		
5E	10	C0	01D2	869		ADDL2    #STACK_SPACE, SP	; restore stack address
			01D5	870			
	54	7C	01D5	871		CLRQ    R4	; R4 and R5 are zero
			01D7	872			; R3 all set
	52	D4	01D7	873		CLRL    R2	; R2 is zero
			01D9	874			; R1 all set
50	56	D0	01D9	875		MOVL    R6, R0	; Set R0 and set condition codes
			01DC	876			; properly
	05	01DC	877			RSB	; to caller

```

01DD 879 .SBTTL CHECK_FOR_FATAL_ERROR
01DD 880 :++
01DD 881 : FUNCTIONAL DESCRIPTION:
01DD 882 :
01DD 883 : This routine looks at current status in R0 and if it finds one of the
01DD 884 : fatal LIB$ errors, its causes the corresponding OTSS error to be
01DD 885 : signalled.
01DD 886 :
01DD 887 :         if it finds:                it signals:
01DD 888 :         -----
01DD 889 :         LIB$_FATERRLIB                OTSS_FATINTERR
01DD 890 :         LIB$_INVSTRDES                OTSS_INVSTRDES
01DD 891 :         LIB$_INSVIRMEM                OTSS_INSVIRMEM
01DD 892 :         LIB$_WRONUMARG                OTSS_WRONUMARG
01DD 893 :         LIB$_INVARG                  OTSS_INVSTRDES
01DD 894 :
01DD 895 :
01DD 896 : CALLING SEQUENCE:
01DD 897 :
01DD 898 :         JMP CHECK_FOR_FATAL_ERROR with INPUT_STATUS.rlc.v in R0
01DD 899 :
01DD 900 : FORMAL PARAMETERS:
01DD 901 :
01DD 902 :         INPUT_STATUS.rlc.v           In R0, the status to be checked.
01DD 903 :                                     On entry, we know it is not a success
01DD 904 :                                     status.
01DD 905 :
01DD 906 : IMPLICIT INPUTS:
01DD 907 :
01DD 908 :         None
01DD 909 :
01DD 910 : IMPLICIT OUTPUTS:
01DD 911 :
01DD 912 :         NONE
01DD 913 :
01DD 914 : ROUTINE VALUE:
01DD 915 : COMPLETION CODES:
01DD 916 :
01DD 917 :         Never returns
01DD 918 :
01DD 919 : SIDE EFFECTS:
01DD 920 :
01DD 921 :         Will signal some error.
01DD 922 :
01DD 923 : --
01DD 924 :
01DD 925 : CHECK_FOR_FATAL_ERROR:
01DD 926 :
01DD 927 :         CMPL    R0, #LIB$_INVSTRDES
01DD 928 :         BNEQ    1$
01DD 929 :         PUSHL   #OTSS_INVSTRDES
01DD 930 :         BRB     FATAL
01DD 931 : 1$:
01DD 932 :         CMPL    R0, #LIB$_INSVIRMEM
01DD 933 :         BNEQ    2$
01DD 934 :         PUSHL   #OTSS_INSVIRMEM
01DD 935 :         BRB     FATAL

```

```

00000000'8F 50 D1 01DD 927
              08 12 01E4 928
00000000'8F DD 01E6 929
              46 11 01EC 930
              01EE 931
00000000'8F 50 D1 01EE 932
              08 12 01F5 933
00000000'8F DD 01F7 934
              35 11 01FD 935

```



```

00000000'8F 50 D1 01FF 936 2$: CMPL R0, #LIB$_FATERRLIB
08 12 0206 937 BNEQ 3$
00000000'8F 24 DD 0208 938 PUSHL #OTSS$_FATINTERR
11 020E 939 BRB FATAL
0210 941 3$:
00000000'8F 50 D1 0210 942 CMPL R0, #LIB$_INVARG
08 12 0217 943 BNEQ 4$
00000000'8F 13 DD 0219 944 PUSHL #OTSS$_INVSTRDES
11 021F 945 BRB FATAL
0221 946 4$:
00000000'8F 50 D1 0221 947 CMPL R0, #LIB$_WRONUMARG
08 12 0228 948 BNEQ 5$
00000000'8F 02 DD 022A 949 PUSHL #OTSS$_WRONUMARG
11 0230 950 BRB FATAL
0232 951 5$:
50 DD 0232 952 PUSHL R0 ; prepare to signal incoming
0234 953 ; error
00000000'GF 01 FB 0234 954 FATAL: CALLS #1, G^LIB$STOP ; to never return
0238 955
0238 956
0238 957
0238 958 .END ; End of module OTSS$COPY

```

OTSS\$COPY  
Symbol table

- String copying module

L 4

16-SEP-1984 00:33:11  
6-SEP-1984 11:15:27

VAX/VMS Macro V04-00  
[LIBRTL.SRC]OTSS\$COPY.MAR;1

Page 23  
(13)

```

CHECK_FOR_FATAL_ERROR      = 000001DD R    03
DESCRIP                    = 00000004
DEST_DESC                  = 0000000C
DSC$A_POINTER              = 00000004
DSC$B_CLASS                = 00000003
DSC$K_CLASS_D              = 00000002
DSC$W_LENGTH               = 00000000
FATAL                      = 00000234 R    03
LEN                        = 00000004
LIB$ANALYZE_SDESC_R2      ***** X    00
LIB$COPY_DXDX6             ***** X    00
LIB$COPY_R_DX6             ***** X    00
LIB$FREE_DD6               ***** X    00
LIB$FREE1_DD6              ***** X    00
LIB$GET1_DD_R6             ***** X    00
LIB$STOP                   ***** X    00
LIB$FATERRLIB              ***** X    00
LIB$INSVIRMEM              ***** X    00
LIB$INVARG                  ***** X    00
LIB$INVSTRDES              ***** X    00
LIB$WRONUMARG              ***** X    00
NUM_DESC                   = 00000004
OTSS$COPY_DXDX             00000067 RG    03
OTSS$COPY_DXDX6            000000B3 RG    03
OTSS$COPY_R_DX             0000013A RG    03
OTSS$COPY_R_DX6            00000172 RG    03
OTSS$FREE_DD               00000027 RG    03
OTSS$FREE1_DD6             0000003A RG    03
OTSS$FREE_DD               00000047 RG    03
OTSS$FREE_DD6              0000005A RG    03
OTSS$GET1_DD               00000000 RG    03
OTSS$GET1_DD_R6            00000017 RG    03
OTSS$FATINTERR             ***** X    00
OTSS$INSVIRMEM             ***** X    00
OTSS$INVSTRDES             ***** X    00
OTSS$WRONUMARG             ***** X    00
SRC_ADDR                   = 00000008
SRC_DESC                   = 00000004
SRC_LEN                    = 00000004
STACK_SPACE                = 00000010
TEMP_DST_ADDR              = 00000004
TEMP_DST_LEN               = 0000000C
TEMP_SRC_ADDR              = 00000000
TEMP_SRC_LEN               = 00000008

```

-----  
! Psect synopsis !  
-----

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 ( 0.)	00 ( 0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
\$ABSS	00000000 ( 0.)	01 ( 1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
_OTSS\$DATA	00000000 ( 0.)	02 ( 2.)	PIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC LONG
_OTSS\$CODE	0000023B ( 571.)	03 ( 3.)	PIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG



```

+-----+
! Performance indicators !
+-----+

```

Phase	Page faults	CPU Time	Elapsed Time
-----	-----	-----	-----
Initialization	34	00:00:00.03	00:00:02.48
Command processing	116	00:00:00.33	00:00:02.20
Pass 1	245	00:00:04.06	00:00:15.93
Symbol table sort	0	00:00:00.58	00:00:02.99
Pass 2	163	00:00:01.55	00:00:06.24
Symbol table output	6	00:00:00.03	00:00:00.03
Psect synopsis output	3	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	569	00:00:06.61	00:00:29.90

The working set limit was 1350 pages.  
37480 bytes (74 pages) of virtual memory were used to buffer the intermediate code.  
There were 40 pages of symbol table space allocated to hold 579 non-local and 35 local symbols.  
958 source lines were read in Pass 1, producing 28 object records in Pass 2.  
11 pages of virtual memory were used to define 9 macros.

```

+-----+
! Macro library statistics !
+-----+

```

<u>Macro library name</u>	<u>Macros defined</u>
_\$255\$DUA28:[SYSLIB]STARLET.MLB;2	5

604 GETS were required to define 5 macros.

There were no errors, warnings or information messages.

MACRO/ENABLE=SUPPRESSION/DISABLE=(GLOBAL,TRACEBACK)/LIS=LIS\$:OTSSCOPY/OBJ=OBJ\$:OTSSCOPY MSRC\$:OTSSCOPY/UPDATE=(ENH\$:OTSSCOPY)

R R R R R R R R R R  
MR MR MR MR MR MR MR MR MR MR



0213 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY

